

Algorithms for data streams

Maria Serna

Spring 2025-2026

- 1 Data stream models
- 2 Graph streams

The data

- Data arrives as sequence of items.
- Sometimes continuously and at high speed.
- Can't store them all in main memory.
- Can't read again; or reading again has a cost.
- Usually data is abstracted to a particular feature of interest, **the label**.
- We discussed some randomized stream algorithms for sampling, according to the uniform distribution, and for approximately counting the number of items.

The computational data stream models

- **Classical streaming model**

- The data stream is accessed sequentially.
- The processing is done sequentially using a small working memory $O(\text{polylog } n)$.
- Measures of complexity: number of **passes** over the data, the size of the working **memory**, the **per-item** processing time.

- **Semi-streaming model**

- Usual for graph problems.
- Working memory is $O(n \text{ polylog } n)$, for a graph with n vertices.
- Enough space to store vertices but not for storing all the edges.

- **Streaming with sorting**

- Allows the creation of intermediate streams.
- Streams can be sorted at no cost.
- Algorithms run in phases reading and creating a stream

Algorithmic goals

- Data streams are potentially of unbounded size.
- As the amount of computation and memory is limited it might be impossible to provide exact answers.
- Algorithms use randomization and seek for an approximate answer.
- Typical approach:
 - Build up a **synopsis data structure**
 - It should be enough to compute answers with a high confidence level.

- 1 Data stream models
- 2 Graph streams**

Streams that describe graphs

- G undirected
- on $[n]$ vertices
- the stream describes the edges of G
- we assume that an edge appears only once in the stream
- We want to keep a DS allowing to answer queries about a graph property
- $O(n \log n)$ memory is reasonable as we are working on the semi-streaming model.

Connectedness

- **Problem:** Decide whether or not the input graph G , given by a stream, is connected.
- **Algorithm:**
 - Maintain a spanning forest H of the seen graph
 - On query answer according to H
- G connected iff admits a spanning tree, the algorithm is correct
- 1 pass, $O(n \log n)$ memory, using a union find DS amortized $O(\alpha(n))$ per item

Maximum matching

- **Problem:** Find a maximum matching in G , given by a stream.
- The algorithm maintains a matching M .
- **Algorithm:**

```
1: procedure MMATCHING(int  $n$ , stream  $s$ , double  $t$ )
2:    $M = \emptyset$ 
3:   while not  $s.end()$  do
4:      $(u, v) = s.read()$ 
5:     if  $M \cup (u, v)$  is a matching then
6:        $M = M \cup \{(u, v)\}$ 
7:   On query, report  $M$ 
```

Maximum matching: Analysis

- 1 pass, $O(n \log n)$ space, $O(1)$ ops. per item
- M is a maximal matching that provides an estimation \hat{f} of the size f of a maximum matching.
- \hat{f} is a 2 approximation to f .

Estimating shortest paths

- **Problem:** Estimate the distance in G , given by a stream, between two given vertices.
- The algorithm builds an intermediate graph H to support queries and uses a parameter t .
- **Algorithm:**

```
1: procedure EDISTANCES(int  $n$ , stream  $s$ , double  $t$ )
2:    $H = ([n], \emptyset)$ 
3:   while not  $s.end()$  do
4:      $(u, v) = s.read()$ 
5:     if  $d_H(u, v) \geq t + 1$  then
6:       add  $(u, v)$  to  $H$ 
7:       update distances
8:   On query  $(u, v)$ , report  $d_H(u, v)$ 
```

Estimating shortest paths: Analysis

- 1 pass, $O(|H| \log n)$ space, $O(n)$ per edge added to H .
- $|H|$?
- Quality?

Estimating shortest paths: Analysis

- A subgraph H is a **t -spanner** for G iff, for all $u, v \in V$ ($V = V(G) = V(H)$),

$$d_G(u, v) \leq d_H(u, v) \leq t d_G(u, v)$$

- Let H be the graph computed by EDISTANCES on (G, t) .

H is a t -spanner.

- H is a subgraph of G , so $d_G(u, v) \leq d_H(u, v)$, for $u, v \in V$.
- When $d_G(u, v) = \infty$ it also holds $d_H(u, v) = \infty$
- Let $x = v_0, v_1, \dots, v_k = y$ be a shortest path from x to y in G .
- Pick $i \in [k]$ and let $e = (v_{i-1}, v_i)$.
If $e \in H$, $d_H(v_{i-1}, v_i) = 1$, otherwise, as e was not added to H , we know that in the current H' when e was read $d_{H'}(v_{i-1}, v_i) \leq t$.
- So, $d_H(x, y) \leq t d_G(x, y)$ as any such H' is a subgraph of H .
- Then, EDISTANCES is a t -approximation

Estimating shortest paths: Analysis

The **girth** $\gamma(G)$ is the length of the shortest cycle in G , ∞ for acyclic G .

Alon, Hoory, Linial 2002

Let n be sufficiently large. Let G be a graph with n vertices and m edges having $\gamma(G) \geq k$, for some integer k . Then

$$m \leq n + n^{1 + \lfloor \frac{k-1}{2} \rfloor}$$

Estimating shortest paths: Analysis

- When $\gamma(G) \geq k$, $m = O(n^{1+2/(k-2)})$.
- By construction the shortest cycle in the subgraph H constructed by EDISTANCES has length $t + 2$.
- Therefore, the memory used is $O(n^{1+2/t} \log n)$.
- We can 3-approximate distances using $O(n^{5/3} \log n)$ memory.